

Flash memory

From Wikipedia, the free encyclopedia
(Redirected from Flash Memory)

Flash memory is non-volatile computer memory that can be electrically erased and reprogrammed. It is a technology that is primarily used in memory cards, USB flash drives (thumb drives, handy drive, memory stick, flash stick, jump drive), which are used for general storage and transfer of data between computers and other digital products. It is a specific type of EEPROM that is erased and programmed in large blocks; in early flash the entire chip had to be erased at once. Flash memory costs far less than byte-programmable EEPROM and therefore has become the dominant technology wherever a significant amount of non-volatile, solid-state storage is needed. Examples of applications include PDAs and laptop computers, digital audio players, digital cameras and mobile phones. It has also gained some popularity in the game console market, where it is often used instead of EEPROMs or battery-powered static RAM (SRAM) for game save data.

Contents

- 1 Overview
- 2 Principles of operation
 - 2.1 NOR flash
 - 2.2 NAND flash
 - 2.3 New designs
- 3 History
- 4 Limitations
- 5 Low-level access
 - 5.1 NOR memories
 - 5.2 NAND memories
 - 5.3 Standardization
- 6 Understanding the distinction between NOR and NAND flash
 - 6.1 Endurance
- 7 Serial flash
 - 7.1 Firmware storage
- 8 Flash file systems
- 9 Capacity
- 10 Speed
- 11 Flash memory as a replacement for hard drives
- 12 See also
- 13 External links
- 14 References

Computer memory types

Volatile

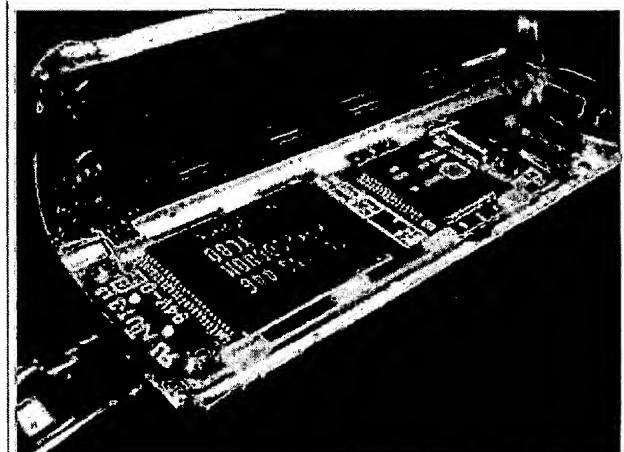
- DRAM
- eDRAM
- SRAM
- 1T-SRAM
- Upcoming
 - Z-RAM
 - TTRAM
- Historical
 - Williams tube
 - Delay line memory
 - Selectron tube

Non-Volatile

- ROM
 - PROM
 - EAROM
 - EPROM
 - EEPROM
 - **Flash memory**
- Upcoming
 - FeRAM
 - MRAM
 - PRAM
 - SONOS
 - RRAM
 - NRAM
- Historical
 - Drum memory
 - Magnetic core memory
 - Bubble memory

Overview

Flash memory is non-volatile, which means that it does not need power to maintain the information stored in the chip. In addition, flash memory offers fast read access times (although not as fast as volatile DRAM memory used for main memory in PCs) and better kinetic shock resistance than hard disks. These characteristics explain the popularity of flash memory for applications such as storage on battery-powered devices. Another feature of flash memory is that when packaged in a "memory card", it is enormously durable, being able to withstand intense pressure, extremes of temperature and immersion in water¹.



A USB flash drive. The chip on the left is the flash memory. The microcontroller is on the right.

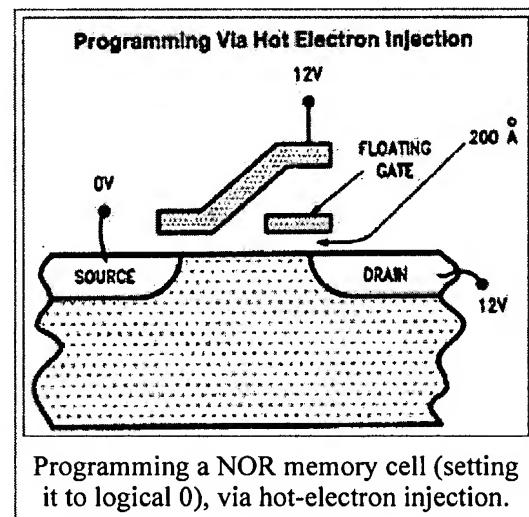
Although technically a type of EEPROM, the term "EEPROM" is generally used to refer specifically to non-flash EEPROM which is eraseable in small blocks, typically bytes. Because an erase cycle is slow, the large size of a flash ROM's erase block can make programming it faster than old-style EEPROM.

Principles of operation

Flash memory stores information in an array of floating-gate transistors, called "cells". In traditional **single-level cell (SLC)** devices, each cell stores only one bit of information. Some newer flash memory, known as **multi-level cell (MLC)** devices, can store more than one bit per cell by choosing between multiple levels of electrical charge to apply to the floating gates of its cells.

NOR flash

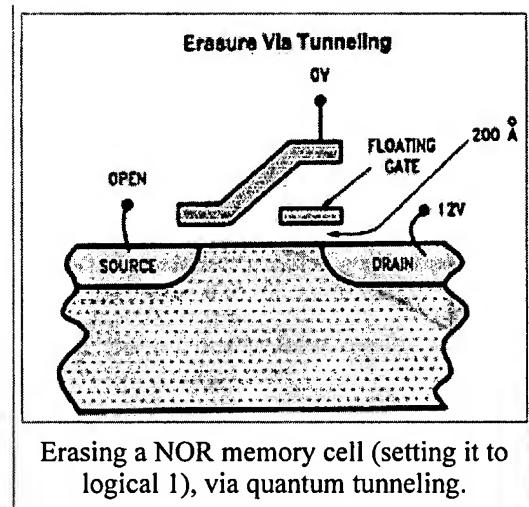
In NOR gate flash, each cell resembles a standard MOSFET, except that it has two gates instead of just one. On top is the control gate (CG), as in other MOS transistors, but below this there is a floating gate (FG) insulated all around by an oxide layer. The FG sits between the CG and the MOSFET channel. Because the FG is electrically isolated by its insulating layer, any electrons placed on it are trapped there and, under normal conditions, will not discharge for a period of many years. When the FG holds a charge, it screens (partially cancels) the electric field from the CG, which modifies the threshold voltage (V_T) of the cell. During read-out, a voltage is applied to the CG, and the MOSFET channel will become conducting or remain insulating, depending on the V_T of the cell, which is in turn controlled by charge on the FG. The presence or absence of current flow through the MOSFET channel is sensed and forms a binary code, reproducing the stored data. In a multi-



level cell device, which stores more than one bit per cell, the *amount* of current flow is sensed (rather than simply its presence or absence), in order to determine more precisely the level of charge on the FG.

A single-level NOR flash cell in its default state is logically equivalent to a binary "1" value, because current will flow through the channel under application of an appropriate voltage to the control gate. A NOR flash cell can be programmed, or set to a binary "0" value, by the following procedure:

- a normal on-voltage (typically 1.5-5 V) is applied to the CG
- the channel is now turned on, so electrons can flow between the source and the drain
- the voltage on the CG is increased, creating an electric field so strong that it sucks electrons through the insulating layer onto the FG, via a process called hot-electron injection



Erasing a NOR memory cell (setting it to logical 1), via quantum tunneling.

To erase a NOR flash cell (resetting it to the "1" state), a large voltage of the *opposite polarity* is applied between the CG and drain, pulling the electrons off the FG through quantum tunneling. Modern NOR flash memory chips are divided into erase segments (often called blocks or sectors). The erase operation can only be performed on a block-wise basis, that is all the cells in an erase segment must be erased together. Programming of NOR cells, however, can generally be performed one byte or word at a time.

Despite the need for high programming and erasing voltages, virtually all flash chips today require only a single supply voltage, and produce the high voltages on-chip via charge pumps.

NAND flash

NAND gate flash uses tunnel injection for writing and tunnel release for erasing. NAND flash memory forms the core of the removable USB interface storage devices known as USB flash drives, as well as most memory card formats available today.

New designs

As manufacturers increase the density of data storage in flash devices, individual memory cells are shrinking and the number of electrons stored in any cell is becoming very small. Coupling between adjacent floating gates can change the write characteristics of cells, making it increasingly difficult to design flash devices for high data integrity. New designs, such as charge trap flash, attempt to provide better isolation between adjacent cells.

History

Flash memory (both NOR and NAND types) was invented by Dr. Fujio Masuoka while working for Toshiba in 1984. According to Toshiba, the name "flash" was suggested by Dr. Masuoka's colleague, Mr. Shoji Ariizumi, because the erasure process of the memory contents reminded him of a flash of a camera. Dr. Masuoka presented the invention at the IEEE 1984 International Electron Devices Meeting

(IEDM) held in San Francisco, California. Intel saw the massive potential of the invention and introduced the first commercial NOR type flash chip in 1988.

NOR-based flash has long erase and write times, but provides full address and data buses, allowing random access to any memory location. This makes it a suitable replacement for older ROM chips, which are used to store program code that rarely needs to be updated, such as a computer's BIOS or the firmware of set-top boxes. Its endurance is 10,000 to 1,000,000 erase cycles. NOR-based flash was the basis of early flash-based removable media; CompactFlash was originally based on it, though later cards moved to less expensive NAND flash.

Toshiba announced NAND flash at ISSCC in 1989. It has faster erase and write times, and requires a smaller chip area per cell, thus allowing greater storage densities and lower costs per bit than NOR flash; it also has up to ten times the endurance of NOR flash. However, the I/O interface of NAND flash does not provide a random-access external address bus. Rather, data must be read on a block-wise basis, with typical block sizes of hundreds to thousands of bits. This makes NAND flash unsuitable to replace program ROM, since most microprocessors and microcontrollers cannot directly execute programs stored in memory without random access; however, NAND flash is similar to other secondary storage devices such as hard disks and optical media, and is thus very suitable for use in mass-storage devices such as memory cards. The first NAND-based removable media format was SmartMedia, and many others have followed, including MultiMediaCard, Secure Digital, Memory Stick and xD-Picture Card. A new generation of memory card formats, including RS-MMC, miniSD and microSD, and Intelligent Stick, feature extremely small form factors; the microSD card has an area of just over 1.5 cm², with a thickness of less than 1 mm.

Limitations

One limitation of flash memory is that although it can be read or programmed a byte or a word at a time in a random access fashion, it must be erased a "block" at a time. This generally sets all bits in the block to 1. Starting with a freshly erased block, any location within that block can be programmed. However, once a bit has been set to 0, only by erasing the entire block can it be changed back to 1. In other words, flash memory (specifically NOR flash) offers random-access read and programming operations, but cannot offer arbitrary random-access rewrite or erase operations. A location can, however, be rewritten as long as the new value's 0 bits are a superset of the over-written value's. For example, a nibble value may be erased to 1111, then written as 1110. Successive writes to that nibble can change it to 1010, then 0010, and finally 0000. Although data structures in flash memory can not be updated in completely general ways, this allows members to be "removed" by marking them as invalid. This technique must be modified somewhat for multi-level devices, where one memory cell holds more than one bit.

Another limitation is that flash memory has a finite number of erase-write cycles (most commercially available flash products are guaranteed to withstand 1000 write-erase-cycles for block 0, and no guarantees for other blocks^[1]). This effect is partially offset by some chip firmware or file system drivers by counting the writes and dynamically remapping the blocks in order to spread the write operations between the sectors; this technique is called wear levelling. Another mechanism is to perform write verification and remapping to spare sectors in case of write failure, which is named bad block management (BBM). With these mechanisms in place, some industry analysts^[2] have calculated that flash memory can be written to at full speed continuously for 51 years before exceeding its write endurance, even if such writes frequently cause the *entire* disk to be overwritten. This figure (51 years) involved a worst-case scenario using specific data parameters and should not be confused with a

particular "shelf life" for a flash memory device. The bottom line is that a typical user using a commercial device, such as a camera, with a flash drive will probably not wear out the memory for the effective life of the camera. However, it - like any other hardware component - can fail. Anyone using flash memory (and any other medium) for critical data would be well advised to backup the data to another device (preferably of a different media).

Low-level access

The low-level interface to flash memory chips usually differs from those of other common types such as DRAM, ROM, and EEPROM, which support random-access via externally accessible address buses.

While NOR memory provides an external address bus for read operations (and thus supports random-access), unlocking, erasing, and writing NOR memory must proceed on a block-by-block basis. Typical block sizes are 64, 128, or 256 bytes. With NAND flash memory, *all* operations must be performed in a block-wise fashion: reading, unlocking, erasing, and writing.

NOR memories

Reading from NOR flash is similar to reading from random-access memory, provided the address and data bus are mapped correctly. Because of this, most microprocessors can use NOR flash memory as execute in place (XIP) memory, meaning that programs stored in NOR flash can be executed directly without the need to copy them into RAM. NOR flash chips lack intrinsic bad block management, so when a flash block is worn out, the software or device driver controlling the device must handle this, or the device will cease to work reliably.

When unlocking, erasing or writing NOR memories, special commands are written to the first page of the mapped memory. These commands are defined by the *Common Flash memory Interface* (CFI) and the flash chips can provide a list of available commands to the physical driver.

Apart from being used as random-access ROM, NOR memories can also be used as storage devices. However, NOR flash chips typically have slow write speeds compared with NAND flash.

NAND memories

NAND flash architecture was introduced by Toshiba in 1989. NAND flash memories cannot provide execute in place due to their different construction principles. These memories are accessed much like block devices such as hard disks or memory cards. The pages are typically 512 or 2,048 bytes in size. Associated with each page are a few bytes (typically 12–16 bytes) that should be used for storage of an error detection and correction checksum.

The pages are typically arranged in blocks. A typical block would be 32 pages of 512 bytes or 64 pages of 2,048 bytes.

While programming is performed on a page basis, erasure can only be performed on a block basis.

NAND devices also require bad block management to be performed by device driver software, or by a separate controller chip (SD cards, for example, include controller circuitry to perform bad block management and wear leveling). When a logical block is accessed by high-level software, it is mapped

to a physical block by the device driver or controller, and a number of blocks on the flash chip are set aside for storing mapping tables to deal with bad blocks.

The error-correcting and detecting checksum will typically correct an error where one bit per 256 bytes is incorrect. When this happens, the block is marked bad in a logical block allocation table, and its undamaged contents are copied to a new block and the logical block allocation table is altered accordingly. If more than one bit in the memory is corrupted, the contents are partly lost, i.e. it is no longer possible to reconstruct the original contents.

Most NAND devices are shipped from the factory with some bad blocks which are typically identified and marked according to a specified bad block marking strategy. By allowing some bad blocks, the manufacturers achieve far higher yields than would be possible if all blocks were tested good. This significantly reduces NAND flash costs and increases the size of the parts.

The first physical block (block 0) is always guaranteed to be readable and free from errors. Hence, all vital pointers for partitioning and bad block management for the device must be located inside this block (typically a pointer to the bad block tables etc). If the device is used for booting a system, this block may contain the master boot record.

When executing software from NAND memories, virtual memory strategies are used: memory contents must first be paged or copied into memory-mapped RAM and executed there. A memory management unit (MMU) in the system is helpful, but this can also be accomplished with overlays. For this reason, some systems will use a combination of NOR and NAND memories, where a smaller NOR memory is used as software ROM and a larger NAND memory is partitioned with a file system for use as a random access storage area. NAND is best suited to flash devices requiring high capacity data storage. This type of flash architecture combines higher storage space with faster erase, write, and read capabilities over the execute in place advantage of the NOR architecture.

Standardization

A group called the Open NAND Flash Interface Working Group (ONFI) has developed a standardized low-level interface for NAND flash chips. This allows interoperability between conforming NAND devices from different vendors. The ONFI specification version 1.0^[3] was released on December 28, 2006. It specifies:

- a standard physical interface (pinout) for NAND flash in TSOP-48, WSOP-48, LGA-52, and BGA-63 packages
- a standard command set for reading, writing, and erasing NAND flash chips
- a mechanism for self-identification (comparable to the Serial Presence Detection feature of SDRAM chips)

The ONFI group is supported by major NAND flash manufacturers, including Intel, Micron Technology, and Sony, as well as by major manufacturers of devices incorporating NAND flash chips. [4]

Understanding the distinction between NOR and NAND flash

NOR and NAND flash differ in two important ways:

- the connections of the individual memory cells are different
- the interface provided for reading and writing the memory is different (NOR allows random-access for reading, NAND allows only block access)

It is important to understand that these two are linked by the design choices made in the development of NAND flash. An important goal of NAND flash development was to reduce the chip area required to implement a given capacity of flash memory, and thereby to reduce cost per bit and increase maximum chip capacity so that flash memory could compete with magnetic storage devices like hard disks.

NOR and NAND flash get their names from the structure of the interconnections between memory cells.

[5] In NOR flash, cells are connected in parallel to the bit lines, allowing cells to be read and programmed individually. The parallel connection of cells resembles the parallel connection of transistors in a CMOS NOR gate. In NAND flash, cells are connected in series, resembling a NAND gate, and preventing cells from being read and programmed individually: the cells connected in series must be read in series.

When NOR flash was developed, it was envisioned as a more economical and conveniently rewriteable ROM than contemporary EPROM, EAROM, and EEPROM memories. Thus random-access reading circuitry was necessary. However, it was expected that NOR flash ROM would be read much more often than written, so the write circuitry included was fairly slow and could only erase in a block-wise fashion; random-access write circuitry would add to the complexity and cost unnecessarily.

Because of the series connection, a large grid of NAND flash memory cells will occupy only a small fraction of the area of equivalent NOR cells (assuming the same CMOS process resolution, e.g. 130 nm, 90 nm, 65 nm). NAND flash's designers realized that the area of a NAND chip, and thus the cost, could be further reduced by removing the external address and data bus circuitry. Instead, external devices could communicate with NAND flash via sequential-access command and data registers, which would internally retrieve and output the necessary data. This design choice made random-access of NAND flash memory impossible, but the goal of NAND flash was to replace hard disks, not to replace ROMs.

Endurance

The endurance of NAND flash is much greater than that of NOR flash (typically 1,000,000 cycles vs. 100,000 cycles). This is because programming and erasure in NOR flash rely on different microscopic processes (hot electron injection and quantum tunneling, respectively), while they are perfectly symmetric in NAND flash (Fowler-Nordheim tunneling).^[5] The asymmetric nature of NOR flash programming and erasure increases the rate at which memory cells degrade, over many program/erase cycles.

The superior symmetric programming method of NAND flash has in fact been adopted in many NOR flash designs, so that some modern NOR chips boast endurance comparable to NAND flash.^[5]

Serial flash

Serial flash is a small, low-power flash memory that uses a serial interface, typically SPI, for sequential data access. When incorporated into an embedded system, serial flash requires fewer wires on the PCB than parallel flash memories, since it transmits and receives data one bit at a time. This may permit a reduction in board space, power consumption, and total system cost.

There are several reasons why a serial device, with fewer external pins than a parallel device, can significantly reduce overall cost:

- Many ASICs are **pad-limited**, meaning that the size of the die is constrained by the number of wire bond pads, rather than the complexity and number of gates used for the device logic. Eliminating bond pads thus permits a more compact integrated circuit, on a smaller die; this increases the number of dies that may be fabricated on a wafer, and thus reduces the cost per die.
- Reducing the number of external pins also reduces assembly and packaging costs. A serial device may be packaged in a smaller and simpler package than a parallel device.
- Smaller and lower pin-count packages occupy reduced PCB area.
- Lower pin-count devices simplify PCB routing.

Firmware storage

With the increasing speed of modern CPUs, parallel flash devices are often too slow to execute in place program code stored on them. Conversely, modern SRAM offers access times below 10 ns, while DDR2 SDRAM offers access times below 20 ns. Because of this, it is often necessary to shadow code stored in flash into RAM; that is, code must be copied from flash into RAM before execution, so that the CPU may access it at full speed. Device firmware may be stored in a serial flash device, and then copied into SDRAM or SRAM when the device is powered-up.^[6] Using an external serial flash device rather than on-chip flash removes the need for significant process compromise (a process that is good for high speed logic is generally not good for flash and vice-versa). Once it is decided to read the firmware in as one big block it is common to add compression to allow a smaller flash chip to be used. Typical applications for serial flash include storing firmware for hard drives, Ethernet controllers, DSL modems, wireless network devices, etc.

Flash file systems

Because of the particular characteristics of flash memory, it is best used with specifically designed file systems which spread writes over the media and deal with the long erase times of NOR flash blocks. The basic concept behind flash file systems is: When the flash store is to be updated, the file system will write a new copy of the changed data over to a fresh block, remap the file pointers, then erase the old block later when it has time. One of the earliest flash file systems was Microsoft's FFS2 (presumably preceded by FFS1), for use with MS-DOS in the early 1990s. Around 1994, the PCMCIA industry group approved the FTL (Flash Translation Layer) specification, which allowed a flash device to look like a FAT disk, but still have effective wear levelling. Other commercial systems such as FlashFX by Datalight were created to avoid patent concerns with FTL.

JFFS was the first flash-specific file system for Linux, but it was quickly superseded by JFFS2, originally developed for NOR flash. Then YAFFS was released in 2003, dealing specifically with NAND flash, and JFFS2 was updated to support NAND flash too. In practice, these filesystems are only used for "Memory Technology Devices" ("MTD"), which are embedded flash memories which do not have a controller. Removable flash media, such as SD and CF cards and USB flash drives, have a controller (often built into the card) to perform wear-levelling and error correction, so use of JFFS2 or YAFFS does not add any benefit. These removable flash memory devices are often used with the old FAT filesystem for compatibility with cameras and other portable devices. Controllerless removable flash memory devices also exist; For example, SmartMedia is even electrically compatible with the

Toshiba TC58 series of NAND flash chips.

Capacity

Common flash memory parts (individual internal components or "chips") range widely in capacity from kilobits to several gigabits each. Multiple chips are often arrayed to achieve higher capacities for use in devices such as the iPod nano or SanDisk Sansa e200. The capacity of flash chips generally follows Moore's law because they are produced with the same processes used to manufacture other integrated circuits. However, there have also been jumps beyond Moore's law due to innovations in technology.

In 2005, Toshiba and SanDisk developed a NAND flash chip capable of storing 1 gigabyte of data using MLC (multi-level cell) technology, capable of storing 2 bits of data per cell. In September 2005, Samsung Electronics announced that it had developed the world's first 2 gigabyte chip.^[7]

In March 2006, Samsung announced flash hard drives with a capacity of 4 gigabytes, essentially the same order of magnitude as smaller laptop hard drives, and in September of 2006, Samsung announced an 8 gigabyte chip produced using a 40 nm manufacturing process.^[8]

For some flash memory products such as memory card and USB-memories, as of mid 2006, 256 megabyte and smaller devices have been largely discontinued. 1 GB capacity flash memory has become the normal storage space for people who do not extensively use flash memory, while more and more consumers are adopting 2 GB, 4 GB, or 8 GB flash drives.

Hitachi (formerly the hard disk unit of IBM) has a competing hard-drive mechanism, the Microdrive, that can fit inside the shell of a type II CompactFlash card. It has a capacity up to 8 GB. BiTMicro offers a 155 GB 3.5" Solid-State disk named the "Edisk".^[9]

Speed

Flash memory cards are available in different speeds. Some are specified the approximate transfer rate of the card such as 2 MB per second, 12 MB per second, etc. The exact speed of these cards depends on which definition of "megabyte" the marketer has chosen to use.

Many cards are simply rated 100x, 130x, 200x, etc. For these cards the base assumption is that 1x is equal to 150 kibibytes per second. This was the speed at which the first CD drives could transfer information, which was adopted as the reference speed for flash memory cards. Thus, when comparing a 100x card to a card capable of 12 MiB per second the following calculations are useful:

$$150 \text{ KiB} \times 100 = 15000 \text{ KiB per second} = 14.65 \text{ MiB per second.}$$

Therefore, the 100x card is 14.65 MiB per second, which is faster than the card that is measured at 12 MiB per second.

Flash memory as a replacement for hard drives

An obvious extension of flash memory would be as a replacement for hard disks. Flash memory does not have the mechanical limitations and latencies of hard drives, so the idea of a solid state drive, or

SSD, is attractive when considering speed, noise, power consumption, and reliability.

There remain some aspects of flash-based SSD's that make the idea unattractive. Most importantly, the cost per gigabyte of flash memory remains significantly higher than that of platter-based hard drives. Although this ratio is decreasing rapidly for flash memory, it is not yet clear that flash memory will catch up to the capacities and affordabilities offered by platter-based storage. Still, research and development is sufficiently vigorous that it is not clear that it will not happen, either.

There is also some concern that the finite number of erase/write cycles of flash memory would render flash memory unable to support an operating system. This seems to be a decreasing issue as warranties on flash-based SSD's are trending to equal or exceed those of current hard drives.^{[10][11]}

As of May 24, 2006, South Korean consumer-electronics manufacturer Samsung Electronics had released the first flash-memory based PCs, the Q1-SSD and Q30-SSD, both of which have 32 GB SSDs.^[12]

At the Las Vegas CES 2007 Summit Taiwanese memory company A-DATA showcased SSD hard disk drives based on Flash technology in capacities of 32 GB, 64 GB and 128 GB.^[13] Sandisk announced an OEM 32 GB 1.8" SSD drive at CES 2007.^[14]

The OLPC XO-1 uses flash memory rather than a hard drive.

As of June 2007, a South Korean company called Mtron claims the fastest SSD with sequential read/write speeds of 100 MB/80 MB per second.^[2] (<http://www.mtron.net/eng/index.asp>)

Rather than entirely replacing the hard drive, hybrid techniques such as hybrid drive and ReadyBoost attempt to combine the advantages of both technologies, using flash as a high-speed cache for files on the disk that are often referenced, but rarely modified, such as application and operation system executable files.

See also

- CompactFlash
- Wear leveling
- DataFlash
- Open NAND Flash Interface Working Group

External links

- Open NAND Flash Interface Working Group (<http://www.onfi.org/index.html>)
- Flash vs. Other Memory Types (<http://www.netrino.com/Publications/Glossary/MemoryTypes.html>)
- A Nonvolatile Memory Overview (<http://aplawrence.com/Makwana/nonvolmem.html>)
- How Flash Memory Works (<http://computer.howstuffworks.com/flash-memory.htm>)
- SanDisk Flash Memory Plant (http://www.pma-show.com/2006/corporate/sandisk_review/001_sandisk_flash_memory_facility.html)
- What is NAND Flash (<http://www.micron.com/products/nand/definingnand>)
- NAND Flash Applications (<http://www.micron.com/products/nand/usingnand>)
- NAND Flash Applications Design Guide

(<http://www.dataio.com/pdf/NAND/Toshiba/NandDesignGuide.pdf.pdf>) from Toshiba (explains the low-level details of interfacing with common NAND flash chips)

References

- Digital Memories Survive Extremes (<http://news.bbc.co.uk/2/hi/technology/3939333.stm>)
- Flash memory database (http://www.letsgodigital.org/en/flash_memory_cards.html)

1. ^ [1] (<http://www.samsung.com/index.htm>)
2. ^ SSD Myths and Legends (<http://www.storagesearch.com/ssdmyths-endurance.html>)
3. ^ http://www.onfi.org/docs/ONFI_1_0_Gold.pdf
4. ^ A list of ONFI members is available at <http://www.onfi.org/onfimembers.html>.
5. ^ ^{a b c} See pages 5-7 of Toshiba's "NAND Applications Design Guide" under External links.
6. ^ Many serial flash devices implement a *bulk read* mode and incorporate an internal address counter, so that it is trivial to configure them to transfer their entire contents to RAM on power-up. When clocked at 50 MHz, for example, a serial flash could transfer a 64 Mbit firmware image in less than two seconds.
7. ^ <http://www.xbitlabs.com/news/memory/display/20050912212649.html>
8. ^ http://tgdaily.com/2006/09/11/samsung_40nm_flash/
9. ^ http://www.tgdaily.com/2005/09/13/bitmicro_rolls_out_155_gig_solid/
10. ^ <http://www.storagesearch.com/semico-art1.html>
11. ^ <http://www.storagesearch.com/bitmicro-art1.html>
12. ^ http://www.samsung.com/he/presscenter/pressrelease/pressrelease_20060524_0000257996.asp
13. ^ Future of Flash revealed (<http://www.theinquirer.net/default.aspx?article=36841>)
14. ^ SanDisk SSD Solid State Drives (<http://www.sandisk.com/Oem/Default.aspx?CatID=1477>)

Retrieved from "http://en.wikipedia.org/wiki/Flash_memory"

Categories: Articles with unsourced statements since April 2007 | All articles with unsourced statements | Articles with unsourced statements since February 2007 | Solid-state computer storage media | Computer memory | Non-volatile memory

- This page was last modified 16:16, 6 July 2007.
- All text is available under the terms of the GNU Free Documentation License. (See [Copyrights](#) for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.

Refine Search

Search Results -

Terms	Documents
L15 and (update or patch).ab.	10

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

L16	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	
<input type="button" value="Refine Search"/>		
<input type="button" value="Recall Text"/>	<input type="button" value="Clear"/>	<input type="button" value="Interrupt"/>

Search History

DATE: Sunday, July 08, 2007 [Purge Queries](#) [Printable Copy](#) [Create Case](#)

<u>Set</u> <u>Name</u>	<u>Query</u>	<u>Hit</u> <u>Count</u>	<u>Set</u> <u>Name</u>
side by side			result set
DB=USPT; PLUR=NO; OP=OR			
<u>L16</u>	L15 and (update or patch).ab.	10	<u>L16</u>
<u>L15</u>	L12 and (update or patch)	75	<u>L15</u>
<u>L14</u>	L12 and L7	0	<u>L14</u>
<u>L13</u>	L12 and L4	0	<u>L13</u>
<u>L12</u>	L11 and firmware	241	<u>L12</u>
<u>L11</u>	ATAPI	714	<u>L11</u>
<u>L10</u>	L4 and (IBM ADJ Technical)	8	<u>L10</u>
<u>L9</u>	L1 and ((Read-only) or (read ADJ only)).ab.	0	<u>L9</u>
<u>L8</u>	L1 and (Read-only or (read ADJ only)).ab.	0	<u>L8</u>
<u>L7</u>	L1 and (Read-only or (read ADJ only))	117	<u>L7</u>
<u>L6</u>	L4 and update.ab.	5	<u>L6</u>
<u>L5</u>	L4 and static.ab.	0	<u>L5</u>
L1 AND (717/168 717/169 717/170 717/171 717/172 717/173 717/174			

<u>L4</u>	717/175 717/176 717/177 717/178).ccls.	56	<u>L4</u>
<u>L3</u>	L1 AND CD.ab.	0	<u>L3</u>
<u>L2</u>	L1 AND ROM.ab.	0	<u>L2</u>
<u>L1</u>	ingberg.xa. or ingberg.xp.	345	<u>L1</u>

END OF SEARCH HISTORY